listening_ window	Defines the time points when the communication window(s) become active (start_time) and inactive (end_time). The start_time implicitly defines the period.
	EXAMPLE When the day of month is not specified (equal to 0xFF) this means that we have a daily listening window management. Daily, monthly ...window management can be defined.
	array window_element
	
	window_element::= structure
	{
	start_time: octet-string,
	end_time: octet-string
	}
	start_time and end_time are formatted as specified in 4.6.1 for *date-time.*

status	Here the status of the window is defined. enum: (0) Inactive: the device will manage no new incoming call. This status is automatically reset to Active when the next listening window starts, (1) Active: the device can answer to the next incoming call, (2) Locked: This value can be set automatically by the device or by a specific client when this client has completed its reading session and wants to give the line back to the customer before the end of the window duration. This status is automatically reset to Active when the next listening window starts.
number_of_ calls	This number is the reference used in modes 1 and 2. When set to 0, this means there is no limit.

number_of_ rings	Defines the number of rings before the meter connects the modem. Two cases are distinguished: The number of rings within the window defined by the attribute *listening_window* and the number of rings outside the *listening_window*.
	nr_rings_type::= structure
	{
	nr_rings_in_window: unsigned (0 = no connection in the window),
	nr_rings_out_of_window: unsigned (0 = no connection outside the window)
	}
	If the number of rings inside and outside the window is the same, the modem always connects regardless of the settings of the *listening_window*.

list_of_ allowed_ callers	Contains an – optional – list of calling numbers which further limits the connectivity of the modem based on the calling number. It also controls the acceptance of wake-up calls or wake-up messages (e.g. SMS) from a calling number.
	
	This requires the presence of a calling line identification (CLI) service in the communication network used.
	
	list_of_allowed_callers::= array list_of_allowed_callers_element
	
	list_of_allowed_callers_element::= structure
	{
	caller_id: octet-string,
	call_type: enum
	}
	
	– the caller_id element holds a calling number from which calls or messages (e.g. SMS) are accepted. The wild-card characters '?' and '*' are supported. With '?' any single character matches, with '*' any character string matches. '*' can only be used at the beginning or at the end of a number, but neither in between nor alone.
	
	Example 1: "+994193500" = only calls from "+994193500" are accepted.
	
	Example 2: "+9941935????" = calls from all numbers in the range of "+99419350000" to "+99419359999" are accepted.
	
	Example 3: "7777*" = calls from all numbers starting with "7777" are accepted.
	
	Example 4: "*9000" = calls from all numbers ending with "9000" are accepted.

list_of_ allowed_ callers (continued)	– the call_type element defines the purpose of the call, i.e. if it's a standard CSD call or a wake-up call / wake-up message. enum: (0) = normal CSD call; the modem only connects if the calling number matches an entry in the list. This is tested in addition to all other attributes, e.g. *number_of_rings*, *listening_windows*, etc. (1) = wake-up request; calls or messages from this calling number are handled as wake-up requests. The wake-up request is processed immediately regardless of all other attributes like *number_of_rings* and *listening_window* (except if the calling number is also present in the list of normal CSD calls, see below). The received message shall be completely empty; otherwise it is not treated as a wake-up message. If the message contains a valid xDLMS APDU from a client in a pre-established AA, the corresponding xDLMS service is executed instead of processing the wake-up request. If the message is not empty but does not contain any valid xDLMS APDU from a client in a pre-established AA then the server does not react. For a call of type (1) the modem *does not* connect – independently of the outcome of the wake-up process. For a call of both type (1) and type (0): see below. If the same calling number is defined as initiator of a normal CSD call (type (0)) and as initiator of a wake-up request (type (1)) the following rule applies for the incoming calls: the wake-up request is only processed if the caller disconnects the line before the *number_of_rings* criterion (depending on *listening _window*) is reached. If the *number_of_rings criterion* is met then a modem connection is established. The *number_of_rings* parameter shall be set large enough to allow the initiator of the call to control the behaviour of the receiver of the call without any knowledge of the time instances of the rings at the receiver's side. NOTE 3 If the *list_of_allowed_callers* is empty (= array [0]) the auto answer function operates in type (0) = normal CSD call and the modem connects independently of the calling number.

5.6.6 Auto connect (class_id = 29, version = 2)

Version 1 of the "Auto connect" class models how the device performs auto dialling or sends messages using various services.

In version 2 new capabilities are added to model the connection of the device to a communication network. Network connection may be permanent, within a time window or on invocation of the connect method.

Auto connect		0...n	class_id = 29, version = 2			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. mode	(static)	enum				x + 0x08
3. repetitions	(static)	unsigned				x + 0x10
4. repetition_delay	(static)	long-unsigned				x + 0x18
5. calling_window	(static)	array				x + 0x20
6. destination_list	(static)	array				x + 0x28
Specific methods		***m/o***				
1. connect (data)		o				x + 0x30

Attribute description	
logical_name	Identifies the “Auto connect” object instance. See 6.2.6.
mode	Controls the auto connect functionality in terms of the timing, the message type and the infrastructure to be used. Modes (1) to (3) are dedicated to CSD services. Modes (4) to (6) are dedicated to sending specific messages using a specific infrastructure. Modes (101) to (104) apply to packet switched network connections only (e.g. GPRS). enum: (0) no auto connect; the device never connects, (1) auto dialling allowed anytime, the values defined in the *calling_window* are ignored, (2) auto dialling allowed within the validity time of the *calling_window*, (3) “regular” auto dialling allowed within the validity time of the *calling_window*; “alarm” initiated auto dialling allowed anytime, (4) SMS sending via Public Land Mobile Network (PLMN), (5) SMS sending via PSTN, (6) email sending, (7...99) reserved, (101) the device is permanently connected to the communication network, (102) the device is permanently connected to the communication network within the validity time of the calling window. The device is disconnected outside the calling window. No connection possible outside the calling window, (103) the device is permanently connected to the communication network within the validity time of the calling window. The device is disconnected outside the calling window but it connects to the communication network as soon as the connect method is invoked,

mode (continued)	(104) the device is usually disconnected. It connects to the communication network as soon as the connect method is invoked, (105…199) reserved, (200…255) manufacturer specific modes.
repetitions	The maximum number of retries in case of unsuccessful connection attempts.
repetition_ delay	The time delay, expressed in seconds until an unsuccessful connection attempt can be repeated. repetition_delay = 0 means delay is not specified

calling_ window	Contains the time points when the window becomes active (start_time), and inactive (end_time). The start_time implicitly defines the period.
	EXAMPLE When the day of month is not specified (equal to 0xFF) this means that the calling window is managed on a daily basis. Daily, monthly …window management can be defined.
	array window_element
	window_element::= structure
	{
	start_time: octet-string,
	end_time: octet-string
	}
	start_time and end_time are formatted as specified in 4.6.1 for *date-time.*

destination_ list	Contains the list of destinations (for example phone numbers, email addresses or their combinations) where the message(s) have to be sent under certain conditions. The conditions and their link to the elements of the array are not defined here. array destination destination::= octet-string
Method description	
connect (data)	Initiates the connection process to the communication network according to the rules defined via the mode attribute. data::= integer (0)

5.6.7 GPRS modem setup (class_id = 45, version = 0)

This IC allows setting up GPRS modems, by handling all data necessary data for modem management.

GPRS modem setup		0…n	class_id = 45, version = 0			
Attributes		*Data type*	*Min.*	*Max.*	*Def.*	Short name
1. logical_name	(static)	octet-string				x
2. APN	(static)	octet-string				x + 0x08
3. PIN_code	(static)	long-unsigned				x + 0x10
4. quality_of_service	(static)	structure				x + 0x18
Specific methods		*m/o*				

Attribute description	
logical_name	Identifies the “GPRS modem setup” object instance. See 6.2.23.
APN	Defines the access point name of the network.
PIN_code	Holds the personal identification number.

quality_of_ service	Specifies the quality of service parameters. It is a structure of 2 elements:
	– the first element defines the default or minimum characteristics of the network concerned. These parameters have to be set to best effort value;
	– the second element defines the requested parameters.
	quality_of_service::= structure
	{
	default: qos_element,
	requested: qos_element
	}
	qos_element::= structure
	{
	precedence: unsigned,
	delay: unsigned,
	reliability: unsigned,
	peak throughput: unsigned,
	mean throughput: unsigned
	}

5.6.8 GSM diagnostic (class_id: 47, version: 1)

The cellular network is undergoing constant changes in terms of registration status, signal quality, etc. It is necessary to monitor and log the relevant parameters in order to obtain diagnostic information that allows identifying communication problems in the network.

An instance of the “GSM diagnostic” class stores parameters of the GSM/GPRS, UMTS, CDMA or LTE network necessary for analysing the operation of the network.

A GSM diagnostic “Profile generic” object is also available to capture the attributes of the GSM diagnostic object, see IEC 62056-6-1:2017, 6.5.

GSM diagnostic		0...n	class_id = 47, version = 1			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. operator	(dyn.)	visible-string				x + 0x08
3. status	(dyn.)	enum	0	255	0	x + 0x10
4. cs_attachment	(dyn.)	enum	0	255	0	x + 0x18
5. ps_status	(dyn)	enum	0	255	0	x + 0x20
6. cell_info	(dyn.)	cell_info_type				x + 0x30
7. adjacent_cells	(dyn.)	array				x + 0x38
8. capture_time	(dyn.)	date-time				x + 0x40
Specific methods		***m/o***				

Attribute description	
logical_name	Identifies the “GSM Diagnostic” object instance. For logical names, see 6.2.23.
operator	Holds the name of the network operator e.g. “YourNetOp”
status	Indicates the registration status of the modem. enum: (0) not registered, (1) registered, home network, (2) not registered, but MT is currently searching a new operator to register to, (3) registration denied, (4) unknown, (5) registered, roaming, (6) ... (255) reserved
cs_attachment	Indicates the current circuit switched status. enum: (0) inactive, (1) incoming call, (2) active, (3) ... (255) reserved
ps_status	The *ps_status* value field indicates the packet switched status of the modem. enum: (0) inactive, (1) GPRS, (2) EDGE, (3) UMTS, (4) HSDPA, (5) LTE, (6) CDMA, (7) ... (255) reserved

cell_info	Represents the cell information:
	cell_info_type::= structure
	{
	cell_ID: double-long-unsigned,
	location_ID: long-unsigned,
	signal_quality: unsigned,
	ber: unsigned,
	mcc: long-unsigned,
	mnc: long-unsigned,
	channel_number: double-long-unsigned
	}
	Where:
	– cell_ID: Four-byte cell ID in hexadecimal format;
	– location_ID: Two-byte location area code (LAC) in the case of GSM networks or Tracking Area Code (TAC) in the case of UMTS, CDMA or LTE networks in hexadecimal format (e.g. "00C3" equals 195 in decimal);

cell_info (continued)	– signal_quality: Represents the signal quality: (0) –113 dBm or less, (1) –111 dBm, (2...30) –109…-53 dBm, (31) –51 or greater, (99) not known or not detectable; – ber: Bit Error Rate (BER) measurement in percent: (0…7) as RXQUAL_n values specified in ETSI GSM 05.08:1996, 8.2.4 (99) not known or not detectable. – mcc: Mobile Country Code of the serving network, as defined in ITU-T E.212 (05.2008); – mnc: Mobile Network Code of the serving network, as defined in ITU-T E.212 (05.2008); – channel_number: Represents the absolute radio-frequency channel number (ARFCN or eaRFCN for LTE network).

adjacent_cells	array adjacent_cell_info
	
	adjacent_cell_info::= structure
	{
	cell_ID: double-long-unsigned,
	signal_quality: unsigned
	}
	Where:
	– cell_ID: Four-byte cell ID in hexadecimal format;
	– signal_quality: Represents the signal quality:
	(0) –113 dBm or less,
	(1) –111 dBm,
	(2...30) –109…-53 dBm,
	(31) –51 or greater,
	(99) not known or not detectable.

capture_time	Holds the date and time when the data have been last captured. *date-time* is formatted as specified in 4.6.1.

5.6.9 LTE monitoring (class_id: 151, version: 0)

Instances of the ‘LTE monitoring” IC allow monitoring LTE modems by handling all data necessary data for this purpose.

LTE monitoring		**0...n**	**class_id = 151, version = 0**			
Attributes		***Data type***	***Min.***	***Max.***	***Def.***	**Short name**
1. logical_name	(static)	octet-string				x
2. lte_quality_of_service	(dyn.)	LTE_qos_type				x + 0x08
Specific methods		***m/o***				

Attribute description	
logical_name	Identifies the “LTE monitoring” object instance. See 6.2.23.

lte_quality_of_ service	Represents the quality of service for the LTE network
	LTE_qos_type::= structure
	{
	T3402: long-unsigned,
	T3412: long-unsigned,
	RSRQ: unsigned,
	RSRP: unsigned,
	qRxlevMin: integer
	}
	Where:
	– T3402: timer in seconds, used on PLMN selection procedure and sent by the network to the modem. Refer to 3GPP TS 24.301 V13.4.0 (2016-01) for details;
	– T3412: timer in seconds used to manage the periodic tracking area updating procedure and sent by the network to the modem. Refer to 3GPP TS 24.301 V13.4.0 (2016-01) for details;
	– RSRQ: represents the signal quality as defined in 3GPP TS 24.301 V13.4.0 (2016-01):
	(0) –19,5dB,
	(1) –19 dB,
	(2...31) –18,5…-3,5 dB,
	(32) –3dB,
	(99) Not known or not detectable;
	– RSRP: represents the signal level as defined in 3GPP TS 24.301 V13.4.0 (2016-01):
	(0) –140dBm,
	(1) –139 dBm,
	(2... 94) –138…-45 dBm,
	(95) –44dBm,
	(99) Not known or not detectable;
	– qRxlevMin: specifies the minimum required Rx level in the cell in dBm as defined in 3GPP TS 24.301 V13.4.0 (2016-01).

5.7 Interface classes for setting up data exchange via M-Bus

5.7.1 Overview

The M-Bus related interface classes specified in this subclause 5.7 are used in two different scenarios:

a) a DLMS/COSEM server hosted by a M-Bus master and exchanging dedicated M-Bus APDUs with M-Bus slaves;

b) a DLMS/COSEM client hosted by a M-Bus master and exchanging DLMS/COSEM APDUs with DLMS/COSEM servers hosted by M-Bus slaves;

In case a) instances of the following M-Bus interface classes are used to set up and manage the M-Bus media in the DLMS/COSEM server:

- M-Bus client (class_id = 72), see 5.7.3;
- M-Bus master port setup (class_id = 74), see 5.7.5;
- M-Bus diagnostic (class_id = 77, version = 0), see 5.7.7.